This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Secure Tracking in Sensor Networks

Chih-Chieh Geoff Chang
North Carolina State University
geoff_chang@ncsu.edu

Wesley E. Snyder
North Carolina State University
wes@ncsu.edu

Cliff Wang
U.S. Army Research Office
cliff.wang@us.army.mil

***Abstract*— Target tracking is a canonical issue in sensor networks research. However, tracking security has gained little or no attention. Once a sensor node is compromised, it will be able to inject false location information into the network, and those nodes receiving such information will suffer greatly in terms of tracking precision. This paper, to the best of our knowledge, is the first to explore the topic of security in the context of Bayesian tracking for sensor networks. We propose to activate more than one nodes at each time step, and use a relaxation labeling algorithm to detect malicious nodes whose reports are then removed. Simulations based on both linear and nonlinear motion models demonstrate that out algorithm works better than simply averaging over the results based on the redundant sets of nodes.**

I. Introduction

Target tracking is a canonical issue in sensor networks research [1]. It is essential to be able to predict the current and future locations of the objects of interest, say, in environmental monitoring or homeland security applications. Target tracking in sensor networks was first proposed to be solved using traditional tracking algorithms such as Bayesian [2], extended Kalman filter [3] or particle filter [4]. Recent research has seen more sophisticated tracking algorithms for sensor networks. For example, [5] and [6] propose distributed particle filters; while [7] proposes decentralized sigma-point information filters for target tracking. Another related and important issue is classification of the motion model of the target(s), and [8] proposes joint multiple-target tracking and classification.

However, tracking security has gained little or no attention. Once one or more of the nodes are compromised, they could generate false measurements or false target-state predictions into the network, and those nodes which receive such information will not be able to produce correct target state estimates. [9] explores a topic similar to tracking security by proposing a protocol to securely verify the time of encounters in multi-hop networks. However, [9] did not address the security issue in the context of Bayesian tracking.

This paper, to the best of our knowledge, is the first paper on secure Bayesian tracking in sensor networks. We propose a new relaxation labeling algorithm to detect malicious nodes. The rest of this paper is organized as follows. Section II describes our system models in terms of tracking. Section III defines the problem, and Section IV proposes a secure-tracking solution. Section V and VI are experimental results and conclusions, respectively.

II. Overview on Target Tracking

A. System Models

A target-tracking problem is to estimate the *states* of the target(s) based on sensor-node measurements. In target tracking, two models form the foundation of all algorithms: the motion model [10] of the target positions and the measurement model of the sensor nodes. Consider the evolution of state sequence $\{\mathbf{x}_k, k \in \mathbb{N}\}$ of a target given by

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \tag{1}$$

where $\mathbf{x}_k$ is the state of the target at time k, f_k is a possibly nonlinear function, $\mathbf{v}_{k-1}$ is an i.i.d. noise sequence, and $\mathbb{N}$ is the set of natural numbers.

The other model of interest is the measurement model of the sensor nodes.Among the various sensor models for different types of sensor nodes, we will use the amplitude sensor model [11] since it models a general amplitude measurement and can be applied to many different types of signals

$$z_k^i = \frac{a}{|\mathbf{x}_k - \mathbf{s}^i|^{\frac{p}{2}}} + n_k \tag{2}$$

where a is the amplitude of the signal emitted from the target and it is assumed to be known. In (2), $\mathbf{s}^i$ is the known location of node i, and p is the attenuation factor.

B. Tracking Algorithm

In a tracking problem, our purpose is to estimate the belief $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ at each time step k. We generally have two stages at each time step k: *prediction* and *update*. Suppose that the required pdf $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ is available. In the prediction stage, we obtain the prior pdf of $\mathbf{x}_k$ via the Chapman-Kolmogorov equation

$$p(\mathbf{x_k}|\mathbf{z_{1:k-1}}) = \int p(\mathbf{x_k}|\mathbf{x_{k-1}})p(\mathbf{x_{k-1}}|\mathbf{z_{1:k-1}})d\mathbf{x_{k-1}} \tag{3}$$

In (3), the probabilistic model of $p(\mathbf{x_k}|\mathbf{x_{k-1}})$ is defined by (1) and the known statistics of $\mathbf{v}_{k-1}$.

As the measurement $\mathbf{z}_k$ becomes available for node i, we may finally obtain the desired $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. This is called the update stage, and Bayes rule is used as

$$p(\mathbf{x_k}|\mathbf{z_{1:k}}) = \frac{p(\mathbf{z_k}|\mathbf{x_k})p(\mathbf{x_k}|\mathbf{z_{1:k-1}})}{p(\mathbf{z_k}|\mathbf{z_{1:k-1}})} \tag{4}$$

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2007**	2. REPORT TYPE	3. DATES COVERED **00-00-2007 to 00-00-2007**
4. TITLE AND SUBTITLE **Secure Tracking in Sensor Networks**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **North Carolina State University,Raleigh,NC,27695**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM002055. Proceedings of the 2007 IEEE International Conference of Communications (ICC 2007) Held in Glasgow, Scotland on June 24-28, 2007. U.S. Government or Federal Rights License

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**6**	

where the normalizing constant

$$p(\mathbf{z_k}|\mathbf{z_{1:k-1}}) = \int p(\mathbf{z_k}|\mathbf{x_k})p(\mathbf{x_k}|\mathbf{z_{1:k-1}})d\mathbf{x_{k-1}} \quad (5)$$

depends on the belief $p(\mathbf{z_k}|\mathbf{x_k})$ defined by (2). The recursive prediction and update relations of (3) and (4) form the optimal Bayesian solution to the tracking problem.

In practice, we use the bootstrap filter algorithm [12] to simulate the prediction and update stages since the the measurement model in (2) and the motion model in (1) can both possibly be nonlinear. We will only briefly describe the bootstrap filter algorithm here, for details and proofs, please refer to [12]. To initiate the algorithm, N samples $\{\mathbf{x}_0(i), i = 1, \cdots, N\}$ are drawn from the known prior, $p(\mathbf{x}_0)$ and $\sum_i \mathbf{x}_0(i) = 1$. At the prediction stage, each sample $\mathbf{x}_{k-1}(i)$ is passed through the system model to obtain samples from the prior at time step k: $\mathbf{x}_k^*(i) = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}(i), \mathbf{v}_{k-1}(i))$, where $\mathbf{v}_{k-1}(i)$ is a sample drawn from the pdf of the nose $p(\mathbf{v}_{k-1})$ in (1).

At the update stage, on receipt of the measurement $\mathbf{z_k}$, we evaluate the likelihood of each prior sample and obtain a normalized weight, q_i, for each sample

$$q_i = \frac{p(\mathbf{z}_k|\mathbf{x}_k^*(i))}{\sum_{j=1}^{N} p(\mathbf{z}_k|\mathbf{x}_k^*(j))} \quad (6)$$

To avoid the degeneracy problem [13] in particle filter methods, we will resample the sample and weight pair $(\mathbf{x}_k^*, q_i)$ for $i = 1, \cdots, N$ times. The resampling procedure is performed by drawing a random sample u_i from the uniform distribution over $(0, 1]$. The value $\mathbf{x}_k^*(M)$ corresponding to

$$\sum_{j=0}^{M-1} q_j < u_i \leq \sum_{j=0}^{M} q_j \quad (7)$$

is selected as a sample for the posterior. Note that in (7), $q_0 = 0$. After the resampling has been done for $i = 1, \cdots, N$ times, we obtain the new samples $\{\mathbf{x}_k(i), i = 1, \cdots, N\}$ so that for any j, $Pr\{\mathbf{x}_k(j) = \mathbf{x}_k^*(i)\} = q_i$.

Note that in particle filter algorithms, the samples $\{\mathbf{x}_k(i), i = 1, \cdots, N\}$ approaches $p(\mathbf{x}_k|\mathbf{z}_k)$ as N asymptotically gets larger.

III. Problem Definition

A. Assumptions

$\langle 1 \rangle$ We assume that malicious nodes can successfully authenticate with the sensor network, and their data can be encrypted with the same key shared with other nodes in the network. $\langle 2 \rangle$ We define malicious nodes to be colluding - they will always report $p(x|z)$ whose mean lies on a incorrect, fictitious path. That is to say, at the same time step, those malicious nodes will point to the same location on the fictitious path. Hence regardless what information from the previous time step is passed to the malicious node, it will produce the false $p(x|z)$ whose mean lies on the fictitious path. The colluding behavior is also what distinguishes malicious nodes from ordinary malfunctional nodes. Figure 1 illustrates a six-node scenario, in which node 3 and node 5 are malicious. $\langle 3 \rangle$ We assume a centralized scenario in which a *central processing unit* will collect tracking reports from the sensor nodes, determine which of them are malicious and remove them. $\langle 4 \rangle$ The purpose of the fictitious path is to allow the enemy to avoid surveillance. However, the fictitious path does not go beyond the sensing range of the nodes or violate the motion model in (1). A rigorous analysis of different behaviors of malicious nodes under the motion model is an ongoing work. $\langle 5 \rangle$ We assume that sensor nodes at successive time steps are positioned so close that they form a clique, i.e. they have unlimited communication bandwidth among each other.

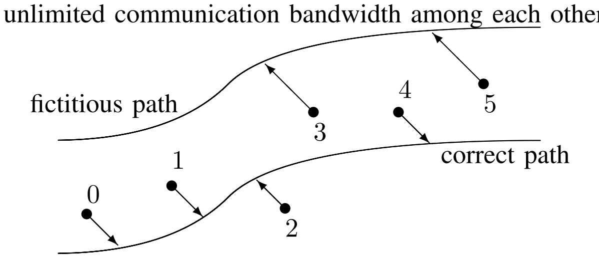


Fig. 1. We have 6 active sensor nodes in the network, which are denoted as 0 through 5. The lower path is the true target path; while the upper one is fictitious. In this scenario, all malicious nodes will report the upper path, i.e. Nodes 3 and 5 are maliciuos

B. Problem Definition

We assume a Bayesian tracking scenario as defined in (3), (4) and (5). Since information from the node at previous time step is crucial in predicting the current target location, activating only one node at a time [2] will have serious security problems. Hence we activate at least two sensor nodes at each time step. Meanwhile, an unknown number of the nodes are malicious and injecting false tracking reports into the network. The problem is to detect those malicious nodes, and to provide a correct target path.

IV. Secure Tracking Algorithm

We propose a secure tracking algorithm based on *relaxation labeling* [14]–[16]. First, we activate two nodes at a time, as illustrated in Figure 2. Let us begin with time step $t = 0$, when the initial position of the target, $p(x_0)$, is assumed to be know. $p(x_0)$ is passed to the two nodes activated at time step $t = 1$. After calculating $p(x_1|z_1)$ using particle filter algorithms, the nodes at $t = 1$ will pass their beliefs to the nodes at $t = 2$. Since each node at $t = 2$ has two $p(x_1|z_1)$ from $t = 1$, it will produce two different $p(x_2|z_2)$. What if, in Figure 2, node 1 is malicious, and node 2 and node 3 are benign? Then the two $p(x_2|z_2)$ calculated by node 3 would be drastically different.

Following such logic, we design our relaxation labeling algorithm based on sets of three nodes, which we denote as *triples*. There are three types of triples:

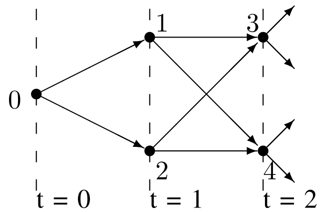


Fig. 2. At $t = 0$, $P(x_0)$ is known, and this information is passed to the two nodes activated at $t = 1$. Using particle filter algorithm, node 1 and node 2 can each calculate $p(x_1|z_1)$, and they are passed to node 3 and node 4. Both node 3 and node 4 have two inputs, hence will produce two distinctive $p(x_2|z_2)$, respectively

1) Type I : a triple consisting of two predecessor nodes and one successor nodes. For example, nodes (1,2,3) and (1,2,4) in Figure 2
2) Type II: a triple consisting of one predecessor node and two successor nodes. For example, nodes (1,3,4) and (2,3,4) in Figure 2
3) Type III: a triple consisting solely of nodes at the same time instant. For example, nodes (1,2,3) and (4,5,6)

In our algorithm, the nodes in a Type III triple do not pass information to each other, hence we will only use Type I and Type II triples in our algorithm.

A. *Type I Triples*

In this section, we focus on Type I triples and how we define the compatibility function for them. In Figure 3(a), we illustrate a Type I triple, and we denote the predecessor nodes as $p1$ and $p2$; and the successor node as s.

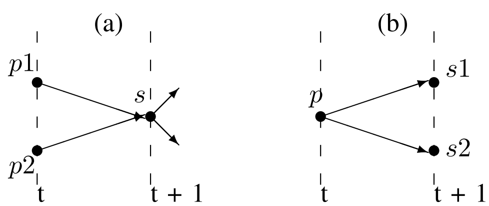


Fig. 3. Two predecessor nodes passing information to one successor node.

Both node $p1$ and node $p2$ pass information to node s, hence node s can examine the difference between the two beliefs that it produces. We denote the belief that node s calculates based on the belief of node $p1$ as $p^1(x|z)$. Similarly, we denote the other belief that node s calculates as $p^2(x|z)$. We can quantify their difference as

$$d = || \int x[p^1(x|z) - p^2(x|z)]dx|| \qquad (8)$$

where $|| \cdot ||$ denotes the Euclidean distance.

If node s is malicious, then regardless of what were passed to it from its predecessors, node s will report a $p(x|z)$ whose mean falls on the fictitious path. Hence d should be close to 0 if node s is malicious. If node s is benign, there are three cases: (i) One of its predecessor is malicious (the other benign). (ii) Both nodes $p1$ and $p2$ are malicious (iii) Both nodes $p1$ and $p2$ are benign. If node s is benign, and one of its predecessor is malicious, the outputs from node s would disagree on each other. Hence we expect d to be large in case (i). However, cases (ii) and (iii) are basically the same, since in case (ii), the two malicious nodes are colluding and both point to the fictitious path. Hence in both case (ii) and (iii), we expect d to be small. We list all the possible cases for a Type I triple in Table I. Note that in Figure 2, the behaviors of Table I apply to nodes $(1, 2, 3)$ and $(1, 2, 4)$.

Predecessor 1	Predecessor 2	Successor	Behavior
malicious	malicious	malicious	$d \simeq 0$
malicious	malicious	benign	d small
malicious	benign	malicious	$d \simeq 0$
malicious	benign	benign	d large
benign	malicious	malicious	$d \simeq 0$
benign	malicious	benign	d large
benign	benign	malicious	$d \simeq 0$
benign	benign	benign	d small

TABLE I

EXPECTED BEHAVIORS IN d FOR A TYPE I TRIPLE

B. *Type II Triples*

In a Type II triple, one predecessor node will pass its $p(x|z)$ to two different successors. We highlight such a scenario in Figure 3(b). We call the predecessor node p, and the two successors node $s1$ and $s2$. Assuming that node p is benign, and one of the successors is malicious (and the other benign), the outputs from the two successor nodes would be different. Again, we can exploit the inconsistency whenever the nodes are behaving differently.

We denote the belief that node $s1$ calculates as $p^1(x|z)$, and the belief that node $s2$ calculates as $p^2(x|z)$. Then we can calculate the difference between $p^1(x|z)$ and $p^2(x|z)$ using (8).

Regardless of whether node p is malicious or not, as long as one successor is malicious and the other is not, we expect d to be large. What if both of node $s1$ and node $s2$ are benign? Since they are given the same input from node p, their outputs should be similar because they are both benign nodes. Hence we expect d to be small when both of node $s1$ and node $s2$ are benign. If both of node $s1$ and node $s2$ are malicious, we expect $d \simeq 0$, since no matter what their inputs are, they will report the fictitious path. We list all 8 possible behaviors in Table II.

C. *Details of the Algorithm*

The expected behaviors in Table I and II are the core of our relaxation labeling algorithm. We define a compatibility function, r, based on d as follows

Predecessor	Successor 1	Successor 2	Behavior
malicious	malicious	malicious	$d \simeq 0$
malicious	malicious	benign	d small
malicious	benign	malicious	d small
malicious	benign	benign	d small
benign	malicious	malicious	$d \simeq 0$
benign	malicious	benign	d large
benign	benign	malicious	d large
benign	benign	benign	d small

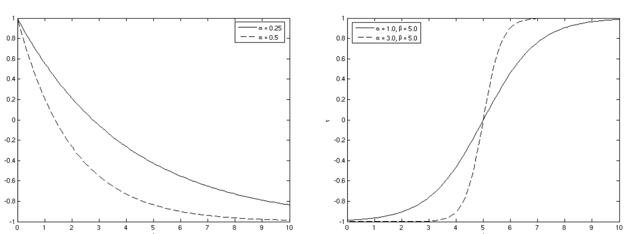
TABLE II

EXPECTED BEHAVIORS IN d FOR A TYPE II TRIPLE

$$r = \begin{cases} 2e^{-\alpha d} - 1, & \text{if } d \text{ small or } 0 \\ \frac{2}{1+e^{-\alpha(d-\beta)}} - 1, & \text{if } d \text{ is large} \end{cases} \tag{9}$$

where α and β are parameters. Note that the compatibility function r will always return a value between -1 and 1. The higher the value that r returns, the more compatible the 3 nodes are. For example, in Figure 2, if we assume node 1 is malicious, node 2 is benign, and node 3 is malicious, then we expect $d \simeq 0$, as in Table I. We then calculate r using $1 - \frac{2}{1+e^{-\alpha(d-\beta)}}$. If the assumption that node 1 is malicious, node 2 is benign, and node 3 is malicious is indeed true, then the empirical data, d, should lead r to return a value close to 1. Otherwise, it should return a negative number which is larger than -1.

We illustrate some parameter settings of (9) in Figure 4. In Figure 4(a), we expect d to be small, hence a small d will lead r to return a value close to 1. On the other hand, we expect d to be large in Figure 4(b), hence a large d will make r return a value close to 1.



(a) r for d small or 0 (b) r for d large

Fig. 4. Illustration of some parameter settings in (9)

So if we assume node 1 is malicious, how is that compatible with, say, node 2 being malicious and node 3 benign? How about node 1 being malicious, node 2 and 3 both being benign? In each possible case, we can calculate the compatibility function r using (9). We denote malicious as λ_0 and benign as λ_1, and we define a function q which accumulates all the effects of labeling node i as λ

$$q_i^t(\lambda) = \frac{1}{N} \sum_j \sum_k \sum_{\lambda'} P_j(\lambda') \sum_{\lambda''} P_k(\lambda'') r(\cdot), \tag{10}$$

where $N = (n-1)(n-2)$, n is the number of nodes in the network, $j = 1, ..., n$, $k = 1, ..., n$, $j \neq i$, $k \neq i$, $j \neq k$, and $P(\cdot)$ is a confidence function to be defined and explained later. Hence in (10), $q_i^t(\lambda)$ is a way to tally all the possibilities when we label node i as λ, label node j as λ' and label node k as λ''. Note that in (10), t is the iteration number, since the relaxation labeling algorithm is an iterative process. Furthermore, in (10), we exclude all the cases that node i, j and k are at the same time step in the tracking process.

Finally, we define the *confidence* $P(\lambda)$. The confidence of node i having label λ_j is denoted as $P_i(\lambda_j)$. The confidence $P_i(\lambda_j)$ has probability-like properties:

$$0 \leq P_i(\lambda_j) \leq 1 \qquad \sum_j P_i(\lambda_j) = 1. \tag{11}$$

Note that in (11), we only have two labels, hence $j = 0, 1$. Following [14], we will iteratively update the confidence of node i having label j as

$$P_i^{t+1}(\lambda_j) = \frac{P_i^t(\lambda_j)\left[1 + q_i^t(\lambda_j)\right]}{D_i^t}, \tag{12}$$

where $D_i^t = \sum_j P_i^t(\lambda_j)\left[1 + q_i^t(\lambda_j)\right]$ is a normalization required to ensure that $P_i(\lambda_j)$ sums to 1, and t stands for iteration. If after many iterations, $P_2^{100}(\lambda_0)$ converges to 1, while $P_2^{100}(\lambda_1)$ converges to 0, then node 2 is found to be malicious.

It is also possible to activate more nodes at each time step, as shown in Figure 5. What is more, if we have more than 3 nodes, we can not only do tracking, but also localization of the target. This will provide added security mechanism using the secure localization algorithms. We summarize the algorithm in Table III.

time t = k	Receive $p(x_{k-1}\|z_{k-1})$
	(if $k == 1$, $p(x_0\|z_0) \equiv p(x_0)$ is known)
	Activate n nodes
	Node i calculates $p^i(x_k\|z_k)$
time t = $k+1$	Activate n nodes
	Each node receives $p^i(x_k\|z_k)$, $i = 1, \ldots, n$
	Node j calculates $p^j(x_{k+1}\|z_{k+1})$, $j = 1, \ldots, n$
	Use relaxation labeling algorithm
	Remove malicious nodes
	Average the results to obtain $\hat{p}(x_{k+1}\|z_{k+1})$
time t = $k+2$	Go to time t = k; use the same algorithm except
	k is replaced with $(k+2)$
	$p(x_{k-1}\|z_{k-1})$ is replaced with $\hat{p}(x_{k+1}\|z_{k+1})$

TABLE III

SECURE TRACKING ALGORITHM USING RELAXATION LABELING

V. EXPERIMENTS

The target is traveling along a one-dimensional space with the following motion model

$$x_t = x_{t-1} + 0.25 + w_{t-1} \tag{13}$$

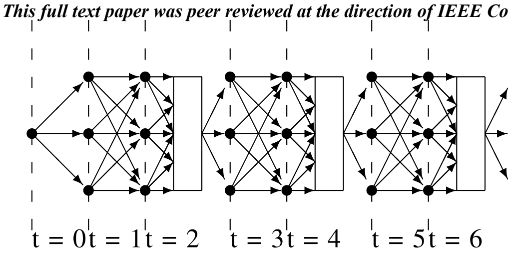
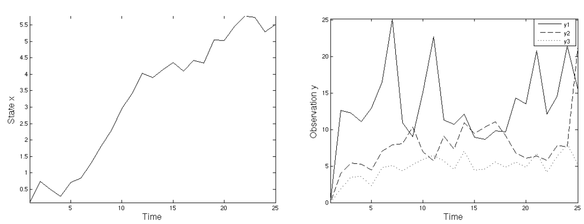


Fig. 5. Illustration of the relaxation labeling algorithm. The rectangular box stands for the central processing unit. At each time step (except time 0), we activate 3 nodes. For every 3 nodes (except time 0), we perform relaxation labeling algorithm to detect malicious nodes. After removing malicious nodes and average the results from benign nodes, the relaxation labeling algorithm produces a correct result and pass it on to the next time step.

where w_{t-1} is the process noise, and its variance is 0.1. The initial position of the target, x_0, is also 0.1. The sensor model is

$$y_t = \frac{20}{|x_t - s_i|} + v_t \tag{14}$$

where v_t is the measurement noise, whose variance is 1.0. We simulate the path of the target over 25 time steps, and it is illustrated in Figure 6(a).



(a) x_t (b) y_t

Fig. 6. System setup for the experiment.

We evenly deploy 15 sensor nodes within $[0, 30]$. At each time step, we will activate 3 sensor nodes. At time $t = k$, we activate the 3 closest nodes to the target position at $t = k - 1$. At time $t = k$, we do not know the current target position until sensor nodes are activated and measurements are made, hence selecting nodes based on the target position at $t = k-1$ is a logical choice. Since we activate 3 nodes at each time step, we denote the closest nodes as set 1, the second-closest nodes as set 2, and so on. We denote s_k^i as the node in set i at time step $t = k$; for example, s_{20}^1 stands for the closest node (to x_{19}) at time step 20. The measurements from the 3 sets of nodes activated are also illustrated in Figure 6(b).

To create attacks, we make an artificial target path which is a line from $x = 20$ at $t = 0$ to $x = 30$ at $t = 25$. We replace one node at each time steps from 5 to 22 with an malicious node which always reports the target position on the fictitious line. The malicious nodes are: s_5^1, s_6^1, s_7^1, s_8^2, s_9^1, s_{10}^3, s_{11}^2, s_{12}^1, s_{13}^2, s_{14}^2, s_{15}^2, s_{16}^3, s_{17}^3, s_{18}^1, s_{19}^3, s_{20}^2, s_{21}^3, s_{22}^3. There is no malicious node at time steps $t \leq 4$ or $t \geq 23$.

After applying relaxation labeling to the 6 nodes at time steps $t = 5, 6$, we show the probability of each node being malicious, $p(\lambda_0)$, in Figure 7. We can clearly see that $p(\lambda_0)$ for the two malicious nodes indeed goes up to 1, while $p(\lambda_0)$ for benign nodes go down to 0. Hence we will remove s_5^1 and s_6^1.

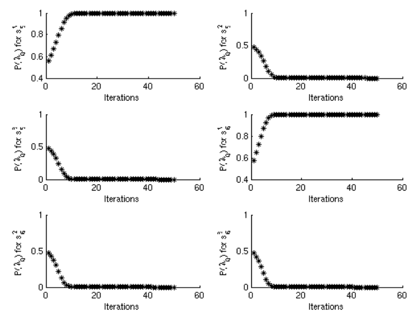


Fig. 7. Probability for being malicious for the 6 nodes at time steps 5 and 6.

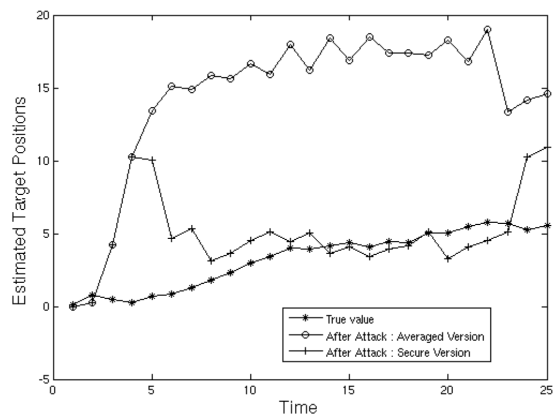


Fig. 8. Tracking performance for activating 3 nodes at each time step. From $t = 5$ to $t = 22$, there is one malicious node at each time step.

Next we apply the relaxation labeling algorithm to $t = 7, 8$, $t = 9, 10, \cdots$ to $t = 21, 22$, and we show the tracking performance in Figure 8. Due to the limitation of space, we do not show the probability of all the nodes (as we did in Figure 7), but the results for all nodes from $t = 5$ to $t = 22$ are all correct.

In order to compare the performance of our algorithm, the tracking result without relaxation labeling is also computed. The 3 sets of nodes work independently. As a result, we obtain 3 separate paths based on 3 different sets of nodes. We then take average of the 3 paths, and the result is shown in Figure 8. The *mean-square error (mse)* for the tracking result using relaxation labeling, as compared to the true target path, is 12.107. On the other hand, the mse for the tracking result without relaxation labeling is 139.293.

As a twist, we repeat our previous experiment, except making two nodes malicious at the same time steps. We select these nodes are malicious nodes: s_6^1, s_6^2, s_8^1, s_8^3, s_{10}^2,

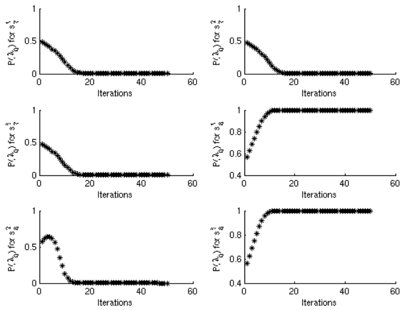


Fig. 9. Probability for being malicious for the 6 nodes at time steps 7 and 8.

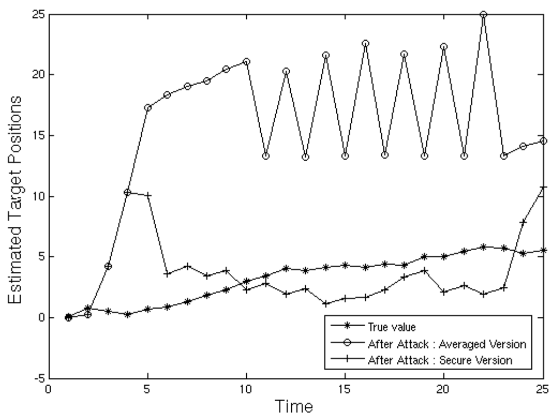


Fig. 10. Tracking performance for activating 3 nodes at each time step. There are two malicious nodes at time steps $t = 6, 8, 10, 12, 14, 16, 18, 20, 22$.

s^3_{10}, s^1_{12}, s^2_{12}, s^1_{14}, s^3_{14}, s^2_{16}, s^3_{16}, s^1_{18}, s^2_{18}, s^1_{20}, s^3_{20}, s^2_{22}, s^3_{22}. After applying relaxation labeling algorithms, we successfully detect all malicious nodes in the network. For example, the result on $t = 7$ and $t = 8$ are illustrated in Figure 9. We can see that the probability of being malicious for s^1_8 and s^3_8 indeed converges to 1. The tracking performance in this experiment is shown in Figure 10. We can observe that although we activate 3 nodes as redundancy in tracking, the malicious nodes still have lasting effects in tracking. Even after we average the results of 3 nodes, we can still see that malicious nodes are successfully at misleading the path, especially at time steps $t = 12, 14, 16, 18, 20, 22$ in Figure 10. On the other hand, the result using relaxation labeling is less effected by the malicious nodes since they are removed. The mse for the 25 time steps is 13.382 with relaxation labeling and 181.803 without relaxation labeling.

VI. Conclusion

We explore a new topic of security in the context of Bayesian tracking in sensor networks. A new algorithm based on relaxation labeling is proposed to detect colluding malicious nodes. We activate at least two nodes at each time step, and inconsistency, if any, between the tracking outputs at successive time steps are exploited. Simulations based on both linear and nonlinear motion models show that out algorithm works better than simply averaging over the results based on the redundant sets of nodes.

Acknowledgment

This project is supported by United States Army Research Office grant W911NF-04-D-003.

References

[1] F. Zhao and L. J. Guibas, *Wireless sensor networks: an information processing approach*. Morgan Kaufmann Publishers, 2004.

[2] J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *EURASIP Journal on Applied Signal Processing*, vol. 4, pp. 378 – 391, March 2003.

[3] R. R. Brooks, C. Griffin, and D. S. Friedlander, "Self-organized distributed sensor network entity tracking," *The International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 207 – 219, Fall 2002.

[4] X. Sheng and Y. H. Hu, "Sequential acoustic energy based source localization using particle filter in a distriuted sensor network," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, May 2004, pp. 17 – 21.

[5] M. Coates, "Distributed particle filters for sensor networks," in *The Third International Symposium on Information Processing in Sensor Networks*, 26 - 27 April 2004, pp. 99 – 107.

[6] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with gmm approximation for multiple targets localization and tracking in wireless sensor network," in *The Fourth International Symposium on Information Processing in Sensor Networks*, 15 April 2005, pp. 181 – 188.

[7] T. Vercauteren and X. Wang, "Decentralized sigma-point information filters for target tracking in collaborative sensor networks," *IEEE Transactions on Signal Processing*, vol. 53, no. 8.2, pp. 2997 – 3009, August 2005.

[8] T. Vercauteren, D. Guo, and X. Wang, "Joint multiple target tracking and classification in collaborative sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 714 – 723, April 2005.

[9] S. Capkun, L. Buttyan, and J.-P. Hubaux, "Sector: secure tracking of node encounters in multi-hop wireless networks," in *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2003, pp. 21 – 32.

[10] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking: dynamic models," in *Proceedings of SPIE Conference on Signal and Data Pocessing of Small Targets*, O. E. Drummond, Ed., vol. 4048, July 2000, pp. 212 – 235.

[11] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *INTERNATIONAL JOURNAL OF HIGH PERFORMANCE COMPUTING APPLICATIONS*, vol. 16, no. 3, Fall 2002.

[12] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear and non-gaussian bayesian state estimation," *IEE Proceedings-F (Radar and Signal Processing)*, vol. 140, no. 2, pp. 107 – 113, April 1993.

[13] M. Arulampalam and S. M. andN. Gordon andT. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174 – 188, February 2002.

[14] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-6, no. 6, pp. 420 – 433, June 1976.

[15] R. A. Hummel and S. W. Zucker, "On the foundations of relaxation labeling processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 3, pp. 267 – 287, May 1983.

[16] J. Kittler and J. Illingworth, "Relaxation labeling algorithms - a review," *Image and Vision Computing*, vol. 3, no. 4, pp. 206 – 216, 1985.